LAUPIES Raphaël

# Multi-Label Classification of Scientific Literature


Aero 4 - Promotion 2026

TF-IDF + Logistic Regression & BERT

June 2025

This report describes and compares two approaches for keyword assignment to scientific articles: a pipeline based on TF-IDF and logistic regression, and a neural approach using BERT. We detail the preprocessing, parameter tuning, results, and comparative insights from both methods, using the NASA SciX corpus as a benchmark. Our focus is on understanding the performance, interpretability, and practical application.

# 1 Data Preparation and Exploration

The task consists of predicting multiple relevant keywords (labels) for each scientific article, given only its title and abstract. We use the SciX dataset, which provides ~18,700 training and ~3,000 validation articles, each annotated with Unified Astronomy Thesaurus (UAT) keywords. The label set is highly imbalanced, ranging from very common to extremely rare terms.

## 1.1 Preprocessing Steps

- Text construction: Title and abstract are concatenated to maximize context for each article.
- Multi-label binarization: The UAT keywords are encoded into a multi-hot vector using `MultiLabelBinarizer`.
- Label filtering: To reduce noise and improve learnability, only labels occurring in at least 10 articles (across train and val) are retained.
- TF-IDF vectorization: For the baseline, input texts are transformed into sparse vectors using TF-IDF (`max_features` in {1000, 2000, 5000}).
- BERT tokenization: For the neural model, texts are tokenized using a pre-trained BERT tokenizer.

Listing 1: Preprocessing steps in code

```
# Concatenate title and abstract
train["full_text"] = train["title"].fillna('') + " " + train["abstract"].fillna('')
# Multi-label binarization
mlb = MultiLabelBinarizer()
mlb.fit(all_labels)
Y_train = mlb.transform(train["verified_uat_labels"])
# Remove rare labels
min_count = 10
label_counts = Y_train.sum(axis=0) + Y_val.sum(axis=0)
labels_to_keep = np.where(label_counts >= min_count)[0]
```

# 2 Model Choices and Parameter Tuning

Our goal was to compare a classical linear approach with a modern neural language model, in order to understand their respective strengths and limitations for large-scale, or sampled, multi-label scientific keyword classification.

## 2.1 TF-IDF + Logistic Regression (Baseline)

**Rationale:** TF-IDF + logistic regression is simple, interpretable, and fast to train, making it an excellent baseline. Each keyword is predicted independently (One-vs-Rest).

**Parameters tuned:**

- TF-IDF `max_features`: We tested 1000, 2000, and 5000. Increasing features allows the model to capture more nuanced vocabulary, but also increases computation time and risk of overfitting.
- Logistic Regression `C`: The inverse regularization parameter. Lower values ($C$ = 0.1) help avoid overfitting, while higher values ($C$ = 1) give the model more capacity. $C$ = 0.5 offered a good trade-off.
- Prediction threshold: By default, a label is predicted if probability > 0.5. In practice, we observed that lower thresholds (0.1–0.15) were needed to increase recall and micro-F1, due to label imbalance.
- Minimum label count: We excluded labels with fewer than 10 examples to avoid noisy and unlearnable classes.

**Pipeline snippet:**

```
vectorizer = TfidfVectorizer(max_features=2000)
X_train = vectorizer.fit_transform(train["full_text"])
clf = OneVsRestClassifier(LogisticRegression(max_iter=300, solver='saga',
n_jobs=-1, C=0.5))
clf.fit(X_train, Y_train)
Y_pred_proba = clf.predict_proba(X_val)
threshold = 0.15
Y_pred = (Y_pred_proba > threshold).astype(int)
```

## 2.2 BERT Neural Network Approach

We included BERT (Bidirectional Encoder Representations from Transformers) because it captures deep semantic and contextual information, which is particularly valuable for scientific language and rare or ambiguous labels. Unlike the baseline, BERT can "understand" the context of entire sentences, rather than just individual word frequency.

Parameters tuned:

- Model: We used `bert-base-uncased` (110M parameters), a widely adopted transformer model for English text.
- Sequence length: 128 tokens, chosen to balance capturing enough context and managing GPU memory usage.
- Batch size: 8, constrained by GPU or CPU memory and chosen to enable efficient training without out-of-memory errors.
- Learning rate: 2e-5, which is a standard starting point for BERT fine-tuning.

- Epochs: Typically 1–3; more epochs can improve results but also increase training time and risk of overfitting.
- Threshold: A value of 0.1 gave the best trade-off between recall and precision for micro-F1, after experimenting with different values (e.g., 0.1–0.3).
- Minimum label count (`min_count`): To avoid noisy and unlearnable labels, only those occurring at least a minimum number of times in the dataset were kept. For experiments on the full dataset (18,677 train, 3,025 val), we set `min_count` = 10 to keep a rich but robust set of labels. For smaller sample-based experiments (1,500 train / 400 val), we increased `min_count` to 30 to avoid extreme imbalance and ensure each label is represented enough for learning. This filtering is critical for stable training and meaningful evaluation.

**Pipeline snippet:**

```
tokenizer = BertTokenizerFast.from_pretrained("bert-base-uncased")
train_dataset = TextDataset(train["full_text"], Y_train)
model = BertForSequenceClassification.from_pretrained(
    "bert-base-uncased", num_labels=num_labels, problem_type="multi_label_classificatio
)
optimizer = AdamW(model.parameters(), lr=2e-5)
# ...training loop...
```

### 2.3 Why These Two Approaches?

- TF-IDF + LogReg gives a quick, robust, and explainable baseline for multi-label tasks. - BERT brings powerful context modeling and can generalize to rare and unseen terms, but is resource-intensive. - Comparing both allows us to analyze the real gains from neural models versus traditional pipelines, and to argue for or against their use in large-scale scientific indexing.

## 3 Results and Comparative Analysis

### 3.1 Baseline (TF-IDF + Logistic Regression)

**Main scores:**

The table summarizes the main evaluation metrics for the multi-label classification task. The columns represent, from left to right:
Precision: the proportion of predicted labels that are actually correct.
Recall: the proportion of true labels that were correctly predicted by the model.
F1-score: the harmonic mean of precision and recall, giving a single measure of overall performance.
Support: the number of true occurrences for each label (or the total number of samples, for averaged scores).

| | | | | |
|---|---|---|---|---|
| micro avg | 0.49 | 0.22 | 0.31 | 13239 |
| macro avg | 0.14 | 0.08 | 0.09 | 13239 |
| weighted avg | 0.30 | 0.22 | 0.23 | 13239 |
| samples avg | 0.41 | 0.25 | 0.27 | 13239 |

Figure 1: Evaluation metrics for TF-IDF + Logisitic Regression (all articles, no seed).

The evaluation metrics highlight the strengths and limitations of the baseline model (TF-IDF + Logistic Regression) for multi-label keyword classification. The Micro-F1 score of 0.31 (31 %) reflects the overall ability of the model to identify the correct keywords, aggregating all predictions and true labels across the entire dataset. In practice, this means that the model successfully recovers about one third of the relevant keywords.

However, the Macro-F1 score is much lower 0.09 (9 %), indicating that performance on rare or highly specific labels is poor. This is a typical outcome in highly imbalanced datasets, where the model tends to focus on predicting frequent keywords and fails to generalize to less common terms.

The Sample-F1 score 0.27 (27 %) shows that, for a given article, the model correctly identifies on average 27% of the true keywords. This metric gives an intuitive sense of how useful the predictions are on an article-by-article basis.

Finally, the micro recall of 0.22 (22 %) reveals that the model retrieves only about 22% of the actual keywords present in the articles. This means it tends to miss many relevant labels, often favoring precision over recall. This behavior is especially pronounced before tuning the prediction threshold: by lowering the threshold (from 0.5 to 0.1–0.15), we observed a significant improvement in recall and the ability to predict more true positives, although at the cost of generating more false positives.

## 3.2 Neural Approach (BERT)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| micro avg | 0.56 | 0.27 | 0.37 | 13239 |
| macro avg | 0.20 | 0.11 | 0.13 | 13239 |
| weighted avg | 0.37 | 0.27 | 0.31 | 13239 |
| samples avg | 0.48 | 0.31 | 0.33 | 13239 |

Figure 2: Evaluation metrics for BERT (all articles, no seed).

The evaluation metrics for the BERT-based model show an improvement compared to the classical baseline. The micro-F1 score rises to 0.37 (37%), indicating a somewhat better overall ability to assign relevant keywords. In practice, this means that BERT recovers about 37% of all true labels, compared to the baseline—an improvement, but not a dramatic leap.

The macro-F1 score also increases to 0.11 (11%), suggesting that BERT is slightly more effective at predicting rare or highly specific keywords. While this gain remains limited, it confirms that BERT's contextual modeling allows for a broader generalization to less frequent terms.

The sample-F1 and micro recall scores benefit similarly, with BERT being more likely to find additional correct labels per article and less prone to missing implicit or context-dependent concepts.

This advantage is relevant for scientific documents, where many important topics are not always stated verbatim.

Overall, while BERT does offer a clear edge over the TF-IDF + logistic regression baseline (particularly for more nuanced or complex classifications) the margin of improvement is relatively modest in this setting. Moreover, these performance gains must be weighed against the considerably greater computational cost and training time required by deep neural models.

### 3.3 Seeded Small-Sample Experiments

To facilitate reproducibility and allow rapid validation, we also evaluated both models on a much smaller subsample: 1500 articles for training and 400 for validation. Here, all random seeds are fixed; running the code with these settings will always produce exactly the same results and figures as below, which makes it easier to check and compare performance without launching long computations.

| | | | | |
|---|---|---|---|---|
| micro avg | 0.70 | 0.07 | 0.12 | 530 |
| macro avg | 0.14 | 0.05 | 0.06 | 530 |
| weighted avg | 0.21 | 0.07 | 0.08 | 530 |
| samples avg | 0.09 | 0.04 | 0.05 | 530 |

Figure 3: Evaluation metrics for TF-IDF + Logistic Regression (sample: 1500/400, fixed seed).

| | | | | |
|---|---|---|---|---|
| micro avg | 0.06 | 0.04 | 0.05 | 530 |
| macro avg | 0.02 | 0.03 | 0.01 | 530 |
| weighted avg | 0.02 | 0.04 | 0.01 | 530 |
| samples avg | 0.05 | 0.03 | 0.03 | 530 |

Figure 4: Evaluation metrics for BERT (sample: 1500/400, fixed seed).

On these smaller data splits, the trend reverses: TF-IDF + Logistic Regression clearly outperforms BERT, with a micro-F1 of 0.12 and a macro-F1 of 0.06 compared to only 0.05 and 0.01 respectively for BERT. This result underlines a key limitation of BERT in low-data regimes: when the number of examples is small, the deep neural approach fails to capture meaningful context and is less robust than the simpler, word-frequency-based TF-IDF baseline.

This seeded sample analysis was specifically included to ensure that anyone running the scripts can reproduce the results shown here without waiting for a full analysis on all the data set. It also highlights an important point: on small samples, TF-IDF remains a better choice, while BERT's strengths only become apparent with much larger datasets.

### 3.4 Detailed Observations

- Label Imbalance: Both models are challenged by highly imbalanced label distribution. On the full dataset, BERT is somewhat better at generalizing to rare or "hidden" keywords, but this advantage largely disappears on small samples, where both approaches struggle.

- Interpretability: The linear model's coefficients can be analyzed directly, whereas BERT's decision process is much less transparent.
- Computational cost: Training BERT is orders of magnitude slower, and brings real gains only on large datasets. On smaller datasets, this cost is not justified by the results.
- Parameter tuning: Micro-F1 is sensitive to the probability threshold; macro-F1 remains stubbornly low, reflecting poor performance on rare labels for both approaches, particularly in the low-data regime.
- Sample size effect: On small datasets, TF-IDF significantly outperforms BERT, as the neural model cannot leverage its contextual capacity without sufficient examples. This behavior is reproducible thanks to the fixed seed in the scripts.

### 3.5 Example Predictions

**Article:** Abundances of Iron-peak Elements in the B8 Mn Star HD 110073
TRUE: `['chemically peculiar stars']`
TF-IDF + LogReg: `[]`

Article: The ALMA Survey of 70 $\mu$m Dark High-mass Clumps...
TRUE: `['interstellar medium', 'molecular clouds', 'star formation']`
TF-IDF + LogReg: `['interstellar medium', 'molecular clouds', 'star formation']`

Article: L-band Calibration of the Green Bank Telescope...
TRUE: `['flux calibration', 'h i line emission', 'radio telescopes', 'surveys']`
TF-IDF + LogReg: `[]`

## 4 Discussion and Perspectives

- The TF-IDF + logistic regression baseline is efficient and explainable, and proves to be robust on small samples, where it even outperforms BERT.
- BERT is much stronger for capturing context and rare labels, but only reveals its potential on large datasets and requires careful tuning, as well as much greater computational resources.
- Label imbalance remains a limiting factor; both models tend to predict frequent, generic keywords and under-predict rare or nuanced ones, with BERT only slightly alleviating this on the full dataset.
- Further improvements could include: data augmentation, hierarchical label modeling, use of domain-specific language models (e.g., SciBERT), or advanced regularization.
- The inclusion of seeded sample experiments in our pipeline allows for rapid reproducibility and evaluation, making it easier for users to check results without rerunning time-consuming full-scale training.

# 5 Conclusion

Both approaches have their strengths: TF-IDF + logistic regression is a fast and transparent baseline that remains highly competitive, especially on smaller datasets. BERT, on the other hand, leverages deep contextual features and can improve classification of complex or rare keywords, but only with sufficiently large training data and at the cost of increased computational time and resource demands. For large-scale scientific information retrieval, neural models like BERT are justified by their better performance, but classical methods still offer a valuable and reliable alternative when data or compute is limited.

# References


- Sadat, M. and Caragea, C. (2022). Hierarchical multi-label classification of scientific documents.
- Liu, R. et al. (2023). Recent advances in hierarchical multi-label text classification: a survey.
- https://huggingface.co/datasets/adsabs/SciX_UAT_keywords
- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- https://medium.com/analytics-vidhya/an-introduction-to-multi-label-text-classification-b1bcb7c